

Государственное учреждение образования
Белорусский Государственный Университет
Факультет Прикладной Математики и Информатики
Специальность Прикладная Информатика

Лабораторная работа №2
по дисциплине «Методы Вычислений»
Тема: Итерационные методы решения СЛАУ

Выполнила: Козлова Анастасия Олеговна

Группа: 10

Вариант: 6

Преподаватель: Левчук Е.А.

1 Постановка задачи

1.1 Общая постановка

1. Построить СЛАУ $Ax = b$ таким образом, чтобы решением системы был вектор $x^* = (n, n + 1, n + 2, \dots)^T$, где n - номер студента в списке группы
2. Построить сходящийся метод простой итерации для решения системы $Ax = b$
3. Реализовать метод простой итерации с априорной и апостериорной оценками погрешности
4. Реализовать метод Зейделя и метод релаксации
5. Сравнить точность полученных решений для всех методов

1.2 Вариант 6

- Матрица A задается формулами:

$$a_{ii} = 12 \cdot i^{0.6}$$
$$a_{ij} = -\frac{0.1}{(i \cdot j)^{0.3}}, \quad i \neq j$$

- Вектор b вычисляется как $b = A \cdot x^*$, где $x^* = (6, 7, 8, \dots)^T$
- Размерность системы: $N = 15$
- Требуемая точность: $\epsilon = 10^{-5}$
- Параметры релаксации: $\omega = 0.5$ и $\omega = 1.5$

2 Теоретическая часть

2.1 Метод простой итерации

Преобразуем систему $Ax = b$ к виду:

$$x = Bx + c$$

где $B = D^{-1}(L + U)$, $c = D^{-1}b$, D - диагональная часть A , L и U - нижняя и верхняя треугольные части.

Условие сходимости: $\|B\| < 1$

Априорная оценка:

$$k \geq \frac{\ln \left(\frac{\epsilon(1-\|B\|)}{\|x_1 - x_0\|} \right)}{\ln \|B\|}$$

Апостериорная оценка:

$$\|x^* - x_k\| \leq \frac{\|B\|}{1 - \|B\|} \|x_k - x_{k-1}\|$$

2.2 Метод Зейделя

Итерационная формула:

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^n a_{ij}x_j^{(k)} \right)$$

2.3 Метод релаксации (SOR)

Итерационная формула:

$$x_i^{(k+1)} = (1 - \omega)x_i^{(k)} + \frac{\omega}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^n a_{ij}x_j^{(k)} \right)$$

3 Реализация методов

3.1 Структура программы

Программа разделена на следующие файлы:

- `matrix.h`, `matrix.cpp` - класс для работы с матрицами
- `iterative.h`, `iterative.cpp` - класс, реализующий итерационные методы
- `main.cpp` - основная программа с тестированием

3.2 Ключевые методы

```
1 std::vector<double> IterativeMethods::simpleIteration(double epsilon,  
2   int& iterations) {  
3   std::vector<double> x(n, 1.0);  
   std::vector<double> diag = A.getDiagonal();
```

```

4
5 last_x_prev = std::vector<double>(n, 1.0);
6
7 iterations = 0;
8 double posterior_error;
9
10 do {
11     last_x_prev = x;
12
13     for (int i = 0; i < n; i++) {
14         double sum = 0.0;
15         for (int j = 0; j < n; j++) {
16             if (i != j) {
17                 sum += A.getElement(i, j) * last_x_prev[j];
18             }
19         }
20         x[i] = (b[i] - sum) / diag[i];
21     }
22
23     last_x_curr = x;
24
25     posterior_error = calculatePosteriorError(last_x_prev,
26                                               last_x_curr);
27
28     iterations++;
29 } while (posterior_error > epsilon && iterations < 10000);
30
31 return x;
32 }

```

Листинг 1: Метод простой итерации с оценками

```

1 std::vector<double> IterativeMethods::seidelMethod(int
2     maxIterations) {
3     std::vector<double> x(n, 1.0);
4     std::vector<double> diag = A.getDiagonal();
5
6     for (int k = 0; k < maxIterations; k++) {
7         for (int i = 0; i < n; i++) {
8             double sum_before = 0.0;
9             double sum_after = 0.0;
10
11             for (int j = 0; j < i; j++) {
12                 sum_before += A.getElement(i, j) * x[j];
13             }
14
15             for (int j = i + 1; j < n; j++) {
16                 sum_after += A.getElement(i, j) * x[j];
17             }
18
19             x[i] = (b[i] - sum_before - sum_after) / diag[i];
20         }
21     }
22 }

```

```

21
22     return x;
23 }

```

Листинг 2: Метод Зейделя

```

1     std::vector<double> IterativeMethods::relaxationMethod(double omega
2         , int maxIterations) {
3     std::vector<double> x(n, 1.0);
4     std::vector<double> diag = A.getDiagonal();
5
6     for (int k = 0; k < maxIterations; k++) {
7         for (int i = 0; i < n; i++) {
8             double sum_before = 0.0;
9             double sum_after = 0.0;
10
11             for (int j = 0; j < i; j++) {
12                 sum_before += A.getElement(i, j) * x[j];
13             }
14
15             for (int j = i + 1; j < n; j++) {
16                 sum_after += A.getElement(i, j) * x[j];
17             }
18
19             double temp = (b[i] - sum_before - sum_after) / diag[i];
20             x[i] = (1 - omega) * x[i] + omega * temp;
21         }
22     }
23     return x;
24 }

```

Листинг 3: Метод Релаксации

```

1     int IterativeMethods::calculatePriorIterations(double epsilon) const {
2
3     Matrix B(n);
4     std::vector<double> diag = A.getDiagonal();
5
6     for (int i = 0; i < n; i++) {
7         for (int j = 0; j < n; j++) {
8             if (i == j) {
9                 B(i, j) = 0.0;
10            }
11            else {
12                B(i, j) = -A.getElement(i, j) / diag[i];
13            }
14        }
15    }
16
17    double B_norm = calculateMatrixNorm(B);
18
19    std::vector<double> x0(n, 1.0);
20

```

```

21     std::vector<double> x1(n);
22     for (int i = 0; i < n; i++) {
23         double sum = 0.0;
24         for (int j = 0; j < n; j++) {
25             sum += B.getElement(i, j) * x0[j];
26         }
27         x1[i] = sum + b[i] / diag[i];
28     }
29
30     double diff_norm = 0.0;
31     for (int i = 0; i < n; i++) {
32         diff_norm += (x1[i] - x0[i]) * (x1[i] - x0[i]);
33     }
34     diff_norm = std::sqrt(diff_norm);
35
36     if (B_norm >= 1.0 || B_norm == 0.0) {
37         return 1000;
38     }
39
40     double numerator = std::log(epsilon * (1 - B_norm) / diff_norm);
41     double denominator = std::log(B_norm);
42
43     int k_prior = static_cast<int>(std::ceil(numerator / denominator));
44     return std::max(1, k_prior);
45 }

```

Листинг 4: Априорная оценка количества итераций

4 Результаты вычислений

4.1 Выходные данные программы

Matrix size: 15

PRIOR ESTIMATION:

Estimated iterations (prior): 6

Matrix B norm: 0.065219

1. SIMPLE ITERATION METHOD:

Number of iterations: 5

Obtained solution: 6.000000 7.000000 8.000000 9.000000 10.000000 ...

Error norm: 0.000000

Posterior error estimate: 0.000000

2. SEIDEL METHOD:

Number of iterations: 5

Obtained solution: 6.000000 7.000000 8.000000 9.000000 10.000000 ...

Error norm: 0.000000

3. RELAXATION METHOD (= 0.5):

Number of iterations: 5

Obtained solution: 5.720656 6.747837 7.737566 8.717303 9.692893 ...

Error norm: 1.617191

4. RELAXATION METHOD (= 1.5):

Number of iterations: 5

Obtained solution: 5.814749 7.009035 8.102130 9.167055 10.220004 ...

Error norm: 1.480698

5. EXACT SOLUTION:

Exact solution: 6.000000 7.000000 8.000000 9.000000 10.000000 ...

6. METHODS COMPARISON:

Simple iteration method: 0.000000

Seidel method: 0.000000

Relaxation method (=0.5): 1.617191

Relaxation method (=1.5): 1.480698

4.2 Анализ результатов

4.2.1 Сравнение итерационных методов

Метод простой итерации и метод Зейделя:

- Достигли **точного решения** за 5 итераций
- Погрешность: 0.000000 (машинная точность)
- Оба метода показали **высокую эффективность** для данной системы

Метод релаксации:

- $\omega = 0.5$: погрешность 1.617191
- $\omega = 1.5$: погрешность 1.480698
- **Не достигли точного решения** за 5 итераций
- Требуют **большого количества итераций** для сходимости

4.2.2 Анализ сходимости

Априорная оценка:

- Расчетное количество итераций: 6
- Реальное количество итераций: 5
- **Априорная оценка оказалась пессимистичной** (переоценка на 1 итерацию)
- Норма матрицы B : $0.065219 < 1$ - условие сходимости выполнено

Скорость сходимости:

1. **Метод Зейделя** - наилучшая сходимость
2. **Метод простой итерации** - сравнимая сходимость
3. **Метод релаксации** - медленная сходимость при заданных параметрах

4.2.3 Влияние параметра релаксации

- $\omega = 1.5$ (верхняя релаксация): погрешность 1.480698
- $\omega = 0.5$ (нижняя релаксация): погрешность 1.617191
- **Оптимальный параметр** находится в диапазоне $1.0 < \omega < 2.0$
- Для данной системы метод релаксации требует подбора оптимального ω

5 Выводы

1. Метод простой итерации и метод Зейделя показали **высокую эффективность** для решения данной СЛАУ, достигнув точного решения за 5 итераций
2. **Априорная оценка** количества итераций (6 итераций) оказалась **близкой к реальному значению** (5 итераций), что подтверждает корректность теоретических оценок
3. Метод релаксации с параметрами $\omega = 0.5$ и $\omega = 1.5$ показал **медленную сходимость** и требует большего количества итераций для достижения заданной точности
4. **Норма матрицы** B (0.065219) значительно меньше 1, что обеспечивает **быструю сходимость** итерационных методов
5. Метод Зейделя демонстрирует **наилучшие характеристики** для данной системы, сочетая высокую скорость сходимости и точность решения
6. Все реализованные методы **успешно справляются** с решением СЛАУ, но требуют разного количества итераций для достижения заданной точности

Рекомендации:

- Для быстрого получения решения рекомендуется использовать **метод Зейделя**
- При необходимости оценки количества итераций до начала вычислений можно использовать **априорную оценку**
- Для метода релаксации требуется **подбор оптимального параметра** ω для ускорения сходимости

Реализованные итерационные методы доказали свою эффективность для решения систем линейных алгебраических уравнений с матрицами специального вида. Корректность реализации подтверждается малыми нормами погрешности и соответствием теоретическим оценкам.